Notes for AA214, Final Projects

T. H. Pulliam

Stanford University

LINEARIZED EULER EQUATIONS

- Project: linearized Euler equations for flow past a thin airfoil.
- Uniform ($\rho = 1, u = M_\infty, v = 0$) at inflow.
- Reference state for the local linearization: uniform flow
- Simplify the equations by assuming constant temperature, i.e. Pressure $= \rho$.

Equations

- The Euler equations for flow about a slender body in two dimensions can be written as (before linearization)

$$\frac{\partial Q}{\partial t} + \frac{\partial E(Q)}{\partial x} + \frac{\partial F(Q)}{\partial y} = 0 \tag{1}$$

with

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \end{pmatrix}, E = \begin{pmatrix} \rho u \\ \rho u^2 + \rho \\ \rho uv \end{pmatrix}, \quad F = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + \rho \end{pmatrix} \tag{2}$$

- *Note:* Q, E, F *are* 3×1 *vectors.*

Quasi-Linear Form

- Rewrite in quasi-linear form, using e.g.

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial Q}\frac{\partial Q}{\partial x} = A\frac{\partial Q}{\partial x}$$

$$\frac{\partial Q}{\partial t} + A\frac{\partial Q}{\partial x} + B\frac{\partial Q}{\partial y} = 0 \tag{3}$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ -u^2+1 & 2u & 0 \\ -uv & v & u \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 0 & 1 \\ -uv & v & u \\ -v^2+1 & 0 & 2v \end{pmatrix} \tag{4}$$

Linearized Form

- Freeze A and B at the reference state $\rho = 1, u = M_\infty, v = 0$

$$\frac{\partial Q}{\partial t} + A\frac{\partial Q}{\partial x} + B\frac{\partial Q}{\partial y} = 0 \tag{5}$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ -M_\infty^2 + 1 & 2M_\infty & 0 \\ 0 & 0 & M_\infty \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & M_\infty \\ 1 & 0 & 0 \end{pmatrix} \tag{6}$$

- The small disturbance form of the Euler equations where ρ, u and v are the pertubation components from a uniform flow in the x direction.
- The Mach number is M_∞ and the equations can be used to study subsonic to supersonic small disturbance flow over slender bodies or past surfaces with small surface variations.

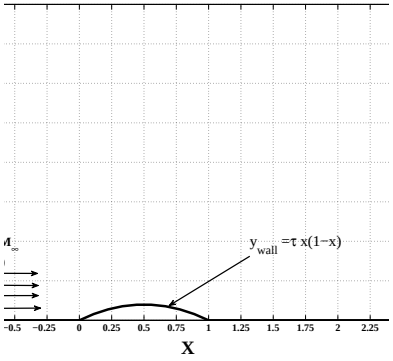
- The matrix A has **real** distinct eigenvalues $M_\infty, M_\infty + 1, M_\infty - 1$ and B the eigenvalues $0, 1, -1$,
- The system is hyperbolic in time.

Geomerty

- The grid is uniform in $-1 \leq x \leq 3.0$ and $0 \leq y \leq 2$
- At the lower surface, a biconvex thin airfoil, τ is the thickness.

$$y_{wall} = \tau x(1-x)/2 \quad 0 \leq x \leq 1 \tag{7}$$

$$y_{wall} = 0 \quad x \leq 0, x \geq 1 \tag{8}$$



Boundary Conditions

- At inflow ($x = -1$):fix $\rho u = M_\infty, \rho v = 0$, set $\frac{\partial \rho}{\partial x} = 0$.
- At the top ($y = 2$):fix all the variables $\rho = 1, u = M_\infty, v = 0$.
- At outflow ($x = 3$): use $\frac{\partial Q}{\partial x} = 0$.
- Assume that v is specified at the lower boundary ($y = 0$) in x using thin airfoil conditions, that is

$$v = M_\infty \frac{dy_{wall}}{dx} \quad \text{imposed at} \quad y = 0 \tag{9}$$

Euler Explicit/Flux Splitting

- Euler explicit time differening ($h = \Delta t$)

$$Q^{(n+1)} = Q^{(n)} + hR(Q)^{(n)} \tag{10}$$

$$R(Q) = -A\frac{\partial Q}{\partial x} - B\frac{\partial Q}{\partial y}$$

- Uniform grid: $x_{j,k} = (j-1)\Delta x$, $y_{j,k} = (k-1)\Delta y$.
- Matries A, B: $\pm$ flux split into A^+, B^+ and A^-, B^-

- New system to be solved with

$$R(Q)^{(n)}_{j,k} = -A^{+}\delta^{b}_{x}Q^{(n)}_{j,k} - A^{-}\delta^{f}_{x}Q^{(n)}_{j,k} \\ -B^{+}\delta^{b}_{y}Q^{(n)}_{j,k} - B^{-}\delta^{f}_{y}Q^{(n)}_{j,k} \qquad (11)$$

- δ^{b}_{x} is a backward differencing operator
- δ^{f}_{x} is a forward differencing operator.
- Equation 10 along with Eq. 11 integrated from uniform initial condition $Q^{(0)} = \begin{pmatrix} 1 \\ M_{\infty} \\ 0 \end{pmatrix}$ to a steady state.

Template Code

- A sample Matlab code is provided for you.
- Euler explicit time differencing with central space differencing.
- Unstable for all CFL
  1. Will run for awhile at low CFL.
  2. Try $CFL = 0.1$ and $nx = 10$
  3. nx points on the airfoil, so $\Delta x = 1.0/(nx - 1)$,
  4. Let $\Delta y = \Delta x$
  5. Produces plots of $C_p = (\rho - 1)/(0.5 * M_\infty^2)$ at the wall
  6. Density contours, and residual history, $||R||_2$.

```
% Euler_template V1.0
%  Final Project
%
% THP 04/2K+1
% Linearized Euler Code for A Thin Circular Arc Airfoil
%
% Flow is Assumed to be rho = 1, u = M_inf , v = 0 at Inflow
% Linearization is about rho = 1, u = M_inf, v = 0
% Thin Airfoil BC
%
% Euler equations are written as
%    ( Pressure = rho is used i.e. Constant Temperature)
%
%   DQ       DQ     DQ
%   --  +  A -- + B --  = 0
%   Dt       Dx     Dy
%
%  where:      |     0         1      0   |       |  rho  |
%         A = | -M_inf^2   2 M_inf   0   |, Q = | rho u |
%             |     0         0    M_inf |       | rho v |
```

```
%
%              |     0          0       1     |
%          B = |     0          0     M_inf   |
%              |     1          0       0     |

%          M_inf :  Free Stream Mach number
%          nx    :  Number of points on airfoil (x=0,1)
%
%  Inputs:
%          cfl   :  dt = min(dx,dy)*cfl/(M_inf+1 + 1)
%                      where Spectral radius of A is (M_inf+1) and B is (1)
%          nmax  : Number of time steps to go
%
clear;
M_inf = 0.8;  % remove comment % $$$ M_inf = input('Enter M_inf = ');

%  Grid Generation
xmin = -1.0;
xmax =  3.0;
ymin =  0.0;
```

```
ymax =  2.0;

nx = 10; % $$$ nx   = input('Enter number of points on body = ');

dx   = 1.0/(nx-1);
dy   = dx;
x    = [xmin:dx:xmax]';
y    = [ymin:dy:ymax]';
jmax = size(x,1);
kmax = size(y,1);
[X,Y] = meshgrid(x,y);

jle = nx;  jte = 2*nx-1;
fprintf(' X of Leading edge point = %g
       X of trailing edge point = %g \n',x(jle),x(jte));

% Surface Definition
tau = 0.1; % tau = input('Enter tau = ');
ys    = zeros(jmax,1);
s     = [jle:jte];
```

```
ys(s) = tau*0.5*x(s).*(1.0-x(s));

% Initialize Q's
Q(:,:,1) = ones(jmax,kmax,1);          % rho
Q(:,:,2) = M_inf*ones(jmax,kmax,1);    % rho u
Q(:,:,3) = zeros(jmax,kmax,1);         % rho v

% Jacobian Matrices
A = zeros(3,3);
B = zeros(3,3);

A(1,2) = 1.0;
A(2,1) = -M_inf^2 + 1;
A(2,2) = 2.0*M_inf;
A(3,3) = M_inf;

B(1,3) = 1;
B(2,3) = M_inf;
B(3,1) = 1;
```

```
%  Thin airfoil BC  v = M_inf D(ys)/Dx
Q(s,1,3) = M_inf*(ys(s+1) - ys(s-1))*0.5/dx;

%  Indices
J  = [1:jmax];
K  = [1:kmax];
JM = [2:jmax-1];
KM = [2:kmax-1];

% Inputs
cfl  = input('Enter cfl = ');
nmax = input('Enter nmax = ');

dt = min(dx,dy)*cfl/(2.0+M_inf);
fprintf(' AT CFL = %g  dt = %g \n',cfl,dt);

for nstep = 1:nmax

  %  rhs Central differencing
  Qx_c(JM,KM,:) = 0.5*(Q(JM+1,KM,:) - Q(JM-1,KM,:))/dx;
```

```
Qy_c(JM,KM,:) = 0.5*(Q(JM,KM+1,:) - Q(JM,KM-1,:))/dy;
for m = 1:3
  R(JM,KM,m) = 0.0;
  for n = 1:3
    R(JM,KM,m) = R(JM,KM,m) + ...
           A(m,n)*Qx_c(JM,KM,n) + ...
           B(m,n)*Qy_c(JM,KM,n) ;
  end
end

% $$$   You can add these next 4 lines and
% $$$         run the code with cfl = 0.1 for 1000 steps
% $$$         to see what a reasonable solution would be.
% $$$   I added this in to give you at a sample working code.
% $$$   Remove % line#
% line1   epse = 0.1;
% line2   Qx_d(JM,KM,:) = (Q(JM+1,KM,:) -2*Q(JM,KM,:) + Q(JM-1,KM,:))/dx;
% line3   Qy_d(JM,KM,:) = (Q(JM,KM+1,:) -2*Q(JM,KM,:) + Q(JM,KM-1,:))/dy;
% line4   R(JM,KM,:) = R(JM,KM,:)-epse*(Qx_d(JM,KM,:) + Qy_d(JM,KM,:));
```

```
resid(nstep) = ...
    (norm(R(:,:,1)) + norm(R(:,:,2)) + ...
                      norm(R(:,:,3)))/(3*jmax*kmax);
if nstep == 1 resid_norm = resid(1);  end
resid(nstep) = resid(nstep)/resid_norm;
fprintf(' nstep = %g  Resid = %g  \n',nstep,resid(nstep));

Q(JM,KM,:) = Q(JM,KM,:) - dt*R(JM,KM,:); % Euler Explicit Step

% BC
% At j = 1 if M-inf < 1 Extrap rho
if M_inf < 1.0
  Q(1,   KM,1) = Q(2,      KM,1);
end
% At jmax DQ/Dx = 0 using backward diff.
Q(jmax,KM,1) = Q(jmax-1, KM,1);
Q(jmax,KM,2) = Q(jmax-1, KM,2);
Q(jmax,KM,3) = Q(jmax-1, KM,3);

% At k=1 D rho/Dy = 0, Du/Dy = 0, fix v
```

```
  RHO = Q(JM,1,1);
  Q(JM,1,1) = Q(JM,2,1);
  Q(JM,1,2) = Q(JM,2,2)./Q(JM,2,1).*Q(JM,1,1);
  % fixing v, not rho v so this little rescaling is necessary
  Q(JM,1,3) = Q (JM,1,3)./RHO.*Q(JM,1,1);

  % At kmax fix all do nothing

end



    % Fix up corners for plot
Q(1,1,:) = Q(2,1,:);
Q(jmax,1,:) = Q(jmax-1,1,:);
Q(1,kmax,:) = Q(2,kmax,:);
Q(jmax,kmax,:) = Q(jmax-1,kmax,:);
figure;
subplot(2,2,1);
surf(X,Y,Q(:,:,1)');
xlabel('x');
```

```
ylabel('y');
zlabel('rho');
view(-25,50);

subplot(2,2,2);
U = Q(:,:,2)./Q(:,:,1);
V = Q(:,:,3)./Q(:,:,1);
surf(X,Y,U');
xlabel('x');
ylabel('y');
zlabel('u');
view(-25,50);


subplot(2,2,3);
P = Q(:,2,1);
cp = (P - 1.0)/(0.5*M_inf^2);
plot(x,-cp);
xlabel('x');
ylabel('-cp');
```

```
subplot(2,2,4);
semilogy([1:nmax],resid);
xlabel('n');
ylabel('resid');

title(['Euler Explicit','  CFL = ',num2str(cfl),...
      '  M_{\infty} = ',num2str(M_inf)]);
```

Project #8,10: Assignment

1. Program the Euler Explicit scheme Eq. 10 with the Flux Split form, Eq. 11 for a uniform grid on the domain ($-1 \leq x \leq 3$, $0 \leq y \leq 2$) using $\tau = 0.1$. Use 1^{st},2^{nd} and 3^{rd} order one-sided differences for $\delta_x^b, \delta_x^f, \delta_y^b$, and δ_y^f.

2. Project #8:Replace the Euler Explicit scheme with Fourth-Order Runge-Kutta and compare the performance and other aspects discussed below.

3. Project # 10:Replace the Euler Explicit scheme with Third-Order Runge-Kutta and compare the performance and other aspects discussed below.

RK4, Project # 8

1. The RK4 scheme is defined as

$$\begin{aligned}
\widehat{Q}^{n+1/2} &= Q^n + \frac{1}{2}hR(Q)^n \\
\tilde{Q}^{n+1/2} &= Q^n + \frac{1}{2}hR(\widehat{Q})^{n+1/2} \\
\overline{Q}^{n+1} &= Q^n + hR(\tilde{Q})^{n+1/2} \\
Q^{n+1} &= Q^n + \frac{1}{6}h\left[R(Q)^n\right. \\
&\qquad +2\left(R(\widehat{Q})^{n+1/2} + R(\tilde{Q})^{n+1/2}\right) + \\
&\qquad \left. R(\overline{Q})^{n+1}\right] \qquad (12)
\end{aligned}$$

2. **Note: the proper use of $\widehat{Q}, \tilde{Q}, \overline{Q}$ is very important!**

RK3, Project # 10

1. The RK3 scheme is defined as

$$\begin{aligned} \widehat{Q} &= Q^{(n)} + \tfrac{1}{4} hR(Q)^{(n)} \\ \tilde{Q} &= Q^{(n)} + \tfrac{8}{15} hR(Q)^{(n)} \\ \overline{Q} &= \widehat{Q} + \tfrac{5}{12} hR(\tilde{Q}) \\ Q^{(n+1)} &= \widehat{Q} + \tfrac{3}{4} hR(\overline{Q}) \end{aligned} \tag{13}$$

2. **Note: the proper use of $\widehat{Q}, \tilde{Q}, \overline{Q}$ is very important!**

Project #8,10 Assignments

1. Discuss the spatial accuracy of this method, e.g., what is er_t or what is the modified wave number? *Hint* Remember, we are working with a linear coupled system. To do the analysis you need to think in terms of the decouple representative equations.

2. Knowing the $\sigma - \lambda$ relation for the Euler explicit and RK4 $O\Delta E$, come up with a stability condition and convergence estimates for different differencing orders. One possible CFL definition for this system is

$$CFL = \frac{h\ (max(\lambda(A)) + max(\lambda(B)))}{\min(dx, dy)}$$

3. Study various CFL numbers (remembering the numerical stability condition!), to see if your analysis is consistent with your results.

4. Pick one of the two below

   (a) Program up at least one more method $O\Delta E$ method and compare it's performance, accuracy and stability with your results above. Look at the other projects for suggestions or just pick one from the textbook or other sources.

i. *Note:* It is not necessary to redo the accuracy and stability analysis for this method, although you may want to make sure it is stable and accurate.)

ii. *Suggestion:* Take a look at Lax-Wendroff (see notes). You could even use an unstable scheme, although I prefer a stable one.

(b) Program the Euler Implicit scheme and compare it's efficiency with the Euler explicit results. You can do this with a central difference implicit operator or an LU operator. See the Linear Euler Equation Project #9, for more details.

Project #9 Euler Implicit / Flux Splitting

- Applying Euler Implicit time differening ($h = \Delta t$)

$$Q^{(n+1)} = Q^{(n)} + hR(Q)^{(n+1)} \tag{14}$$

$$R(Q) = -A\frac{\partial Q}{\partial x} - B\frac{\partial Q}{\partial y}$$

- Discretize the field using a uniform grid with $x_{j,k} = (j-1)\Delta x$ and $y_{j,k} = (k-1)\Delta y$.
- Matries A, B can be $\pm$ flux split into A^+, B^+ and A^-, B^- as discussed in class.
- New system to be solved with

$$\begin{aligned} R(Q)_{j,k}^{(n+1)} = &-A^+\delta_x^b Q_{j,k}^{(n+1)} - A^-\delta_x^f Q_{j,k}^{(n+1)} \\ &-B^+\delta_y^b Q_{j,k}^{(n+1)} - B^-\delta_y^f Q_{j,k}^{(n+1)} \end{aligned} \tag{15}$$

where e.g., δ_x^b is a backward differencing operator and δ_x^f is a forward differencing operator.

- Equation 14 along with Eq. 15 can be integrated from the uniform initial condition $Q^(0) = \begin{pmatrix} 1 \\ M_\infty \\ 0 \end{pmatrix}$ to a steady state.

Assignment, Project #9

1. Program the Euler Implicit scheme Eq. 14 with the Flux Split form, Eq. 15 for a uniform grid on the domain ($-1 \leq x \leq 3$, $0 \leq y \leq 2$) using $\tau = 0.1$. Use 1^{st} order one-sided differences for $\delta_x^b, \delta_x^f, \delta_y^b$, and δ_y^f.

2. The system is already in linearized form so that the implicit operators can be defined in a number of ways, e.g., keeping the x operators together results in the large two-dimensional implicit matrix operator, or one could factor the x and y operators or do an upper-lower factorization. Pick an approach for the implicit operator and program it up.

Some suggested choices are:

(a) Unfactored: leads to the large two-dimensional implicit operator. This is the most work intensive choice, it could be fun.

(b) Block-Tri-Diagoanal Factored: Keep A^+ and A^- terms together and B^+ and B^- terms together, i.e., this produces the standard spatial factorization of x and y.

(c) Lower-Upper Facrotization: Keep A^+ and B^+ terms together and A^- and B^- terms together producing Lower-Block-Tri-Diagonal and Upper-Block-Tri-Diagonal Operators.

3. Find the expression for the $\sigma - \lambda$ relation for your choosen method. What is the accuracy of this method, i.e., what is er_λ? *hint* Remember, we are working with a linear coupled system. To do the analysis you need to think in terms of the decouple representative equations.

4. One possible CFL definition for this system is

$$CFL = \frac{h\ (max(\lambda(A)) + max(\lambda(B)))}{\min(dx, dy)}$$

Study various CFL numbers. What type of stability do you see? Is your analysis consistent with your results?

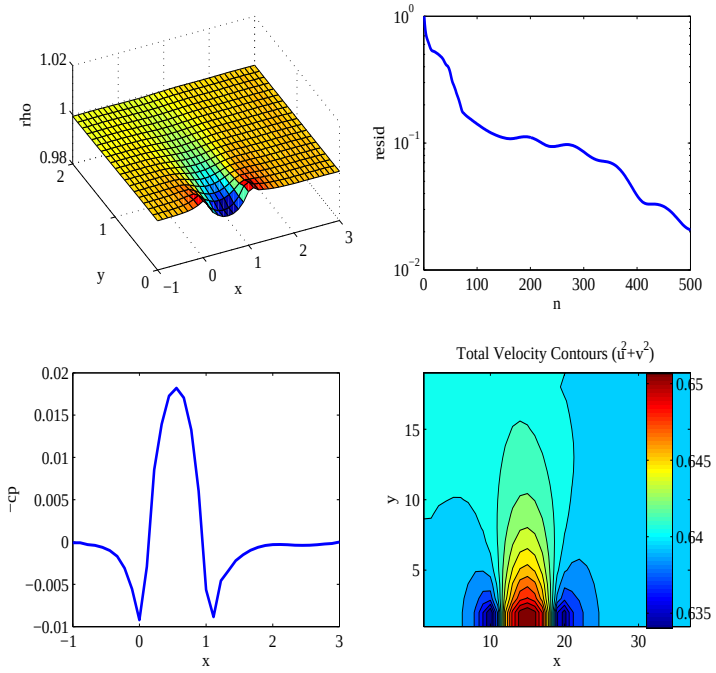
5. Program up at least one more method $O\Delta E$ method and compare it's performance, accuracy and stability with your results above. Look at the other projects for suggestions or just pick one from the textbook or other sources.

   (a) *Note:* It is not necessary to redo the accuracy and stability analysis for this method, although you may want to make sure it is stable and accurate.)

   (b) *Suggestion:* Choose one of the other possibilities I gave you above. How does it compare in terms of stability or efficiency.

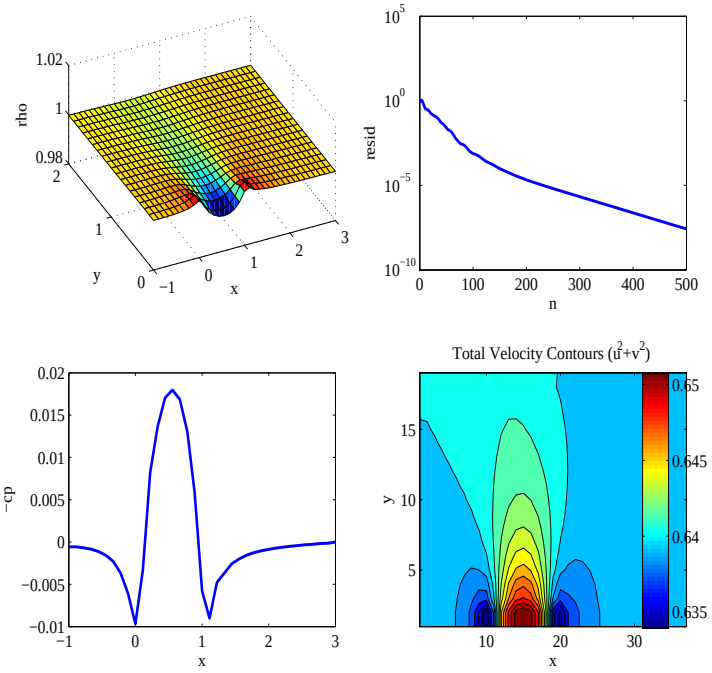
PROJECT #8: Example

Euler2D:: RK4 CFL = 0.8 M_∞ = 0.8 3rd O Upwind Simple BC



PROJECT #9: Example

Euler2D:: Delta AF Implicit X−Y CFL = 10 M_∞ = 0.8 3rd O Upwind Simple BC



PROJECT #11: NONLINEAR EULER EQUATIONS

- Project solves the unsteady Euler equations for vortex propagation.
- The initial condition is a finite core vortex which will propagated from left to right at a fixed speed, M_∞.
- The domain is periodic in x, so boundary conditons a simple.
- The Euler equations in two dimensions can be written as

$$\frac{\partial Q}{\partial t} + \frac{\partial E(Q)}{\partial x} + \frac{\partial F(Q)}{\partial y} = 0 \tag{16}$$

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \end{pmatrix}, E = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (e+p)u \end{pmatrix}, \quad F = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (e+p)v \end{pmatrix} \tag{17}$$

with the equation of state for pressure is
$p = ((\gamma - 1))(e - \frac{1}{2}\rho(u^2 + v^2))$, temperature $T = \frac{p}{\rho}$ and $\gamma = 1.4$.

- *Note:* Q, E, F *are* 4×1 *vectors.*

Problem Setup

- Geometry: Uniform grid:: $0 \leq x \leq 10$ and $0 \leq y \leq 10$.
- Boundary conditions: Periodic in x and y. Note the domain is rectangular with limits $(0, 10)$, but the flow is assumed to be periodic over those limits.
- Initial conditions and exact solution:
- A finite core vortex embedded in a free stream flow $Q_\infty = (\rho_\infty, M_\infty, 0, e_\infty)$
- With $\rho_\infty = 1.0, M_\infty = 0., p_\infty = 1.0$, and $T_\infty = 1.0$.
- The perturbed (by the vortex) field is

$$T = T_\infty - \frac{\alpha^2(\gamma - 1)}{16\phi\gamma\pi^2} e^{2\phi(1-r^2)} \tag{18}$$

$$\rho = T^{\frac{1}{(\gamma-1)}} \tag{19}$$

$$u = M_\infty - \frac{\alpha}{2\pi}(y - y_0)e^{\phi(1-r^2)} \tag{20}$$

$$v = \frac{\alpha}{2\pi}(x - x_0)e^{\phi(1-r^2)} \tag{21}$$

- α the vortex strength (use $\alpha = 5.0$) and $\phi = 0.5$ is the gaussian width scale.
- The vortex is initally centered at $x_0 = 5.0$ and $y_0 = 5.0$ and $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$

Apply the Euler Explicit

- Euler explicit time differening ($h = \Delta t$),

$$Q^{(n+1)} = Q^{(n)} + hR(Q)^{(n)} \tag{22}$$

$$R(Q)^{(n)} = -\frac{\partial E^n}{\partial x} - \frac{\partial F^n}{\partial y} \tag{23}$$

- Use nx, # points in x and y (keep the aspect ratio square)
- Let $\Delta x = \Delta y = \frac{10.0}{nx-1}$.
- Discretize the field using a uniform grid with $x_{j,k} = (j-1)\Delta x$ and $y_{j,k} = (k-1)\Delta y$, for $j = 1, 2, 3, \ldots nx - 1$ and $k = 1, 2, 3, \ldots nx - 1$.
- Note the computational domain falls short of 10 by Δx and Δy, in x and y respectively, since e.g. $x = 0$ and $x = 10.0$ are the same location because of the periodicity conditions.

Flux Splitting Method in space.

- Flux Jacobians are $A = \frac{\partial E}{\partial Q}$ and $B = \frac{\partial F}{\partial Q}$.
- The matries A, B can be $\pm$ flux split into A^+, B^+ and A^-, B^- as discussed in class.
- New split fluxes are defined as $E^\pm = A^\pm Q$ and $F^\pm = B^\pm Q$.
- This produces the new system to be solved with

$$R(Q)_{j,k}^{(n)} = -\delta_x^b (E^+)_{j,k}^{(n)} - \delta_x^f (E^-)_{j,k}^{(n)} \\ -\delta_y^b (F^+)_{j,k}^{(n)} - \delta_y^f (F^-)_{j,k}^{(n)} \qquad (24)$$

- δ_x^b: backward differencing operator, δ_x^f:forward differencing.

Sample Project Code

- A sample Matlab code is provided for you.
- Euler explicit time differencing with central space differencing.
- Unstable for all $CFL = \frac{\Delta t M_\infty}{\Delta x}$, but will run for awhile at low CFL. Try $CFL = 0.1$ and $nx = 21$.
- Produces plots of the vortex as it propagates from left to right, through the right boundary reappearing at the left boundary and continuing on.
- You can pick the number of revolutions of the vortex and for an inputted CFL compute the number of time steps to complete the revolutions.

```
% Euler_vortex V1.0
%  Final Project
%
% THP 11/06/2006
%
%
% Flow is Assumed to be a compresible vortex
%     see: vortex_exact.m for details.
%
% Euler equations are written as
%
%   DQ        DE      DF
%   --   +    --   +  --   = 0
%   Dt        Dx      Dy
%
%  where:      | rho u        |          |  rho   |
```

```
%          E = | rho u^2 + p |,   Q = | rho u |
%              | rho u v     |         | rho v |
%              | (e + p) u   |         |   e   |
%
%  where:      | rho v       |
%          F = | rho u v     |,  p = (gamma-1)*(e-0.5*rho(u^2 v^2))
%              | rho v^2 + p |
%              | (e + p) v   |   gamma = 1.4
%
%  Inputs:
%          nx     :  Number of points accross domain in x and
%
%          cfl    :  dt = min(dx,dy)*cfl/(M_inf)  M_inf: Mach umber
%
%          revolutions  : Number of passes of vortex around p riodic
%
```

```
clear;
fig1 = 0;
fig2 = 0;
M_inf = 0.5;

gamma = 1.4;
gm1   = (gamma-1);
%  Grid Generation
xmin =  0.0;
xmax =  10.0;
ymin =  0.0;
ymax =  10.0;

nx = 21;
ss = ['Enter  nx Default = ' num2str(nx) ' : '];
```

```
bb = input(ss); if size(bb) ~= 0 nx = bb; end

dx   = (xmax-xmin)/(nx-1);
dy   = (ymax-ymin)/(nx-1);;
x    = [xmin:dx:xmax-dx]';
y    = [ymin:dy:ymax-dy]';
jmax = size(x,1);
kmax = size(y,1);
[X,Y] = meshgrid(x,y);

time = 0;
ivar = 1;
pvar = 1;
x0 = 5.0;  y0 = 5.0;
Q = vortex_exact(M_inf,0.0,X',Y',jmax,kmax,x0,y0);
```

```
QE = Q;
rho_ex_min = min(min(Q(:,:,1)));

% Jacobian Matrices
A = zeros(jmax,kmax,4,4);
B = zeros(jmax,kmax,4,4);



%  Indices
J  = [1:jmax];
K  = [1:kmax];
JM = [2:jmax-1];
KM = [2:kmax-1];

JPLUS = J+1;
JMINUS = J-1;
```

```
JPLUS(jmax) = 1;
JMINUS(1) = jmax;
KPLUS = K+1;
KMINUS = K-1;
KPLUS(kmax) = 1;
KMINUS(1) = kmax;

cfl = 0.2;
ss = [ 'Enter  cfl Default = ' num2str(cfl) ' : '];
bb = input(ss); if size(bb) ~= 0 cfl = bb; end

dt = cfl*dx/(M_inf);
fprintf(' AT CFL = %g  dt = %g \n',cfl,dt);

revolutions = 0.1;
```

```
ss = [ 'Enter  revolutions Default = ' num2str(revolutions)   : '];
bb = input(ss); if size(bb) ~= 0 revolutions = bb; end



nmax = floor((10.0)*revolutions/dt/M_inf);
disp([' Nmax = ' num2str(nmax)]);

for nstep = 1:nmax
  time = nstep*dt*M_inf;

   rho = Q(:,:,1);
   u = Q(:,:,2)./rho;
   v = Q(:,:,3)./rho;
   e = Q(:,:,4);
   P = gm1*(e-0.5*rho.*(u.^2+v.^2));
```

```
E(:,:,1) = Q(:,:,2);
E(:,:,2) = Q(:,:,2).*u + P;
E(:,:,3) = Q(:,:,3).*u;
E(:,:,4) = (e+P).*u;
F(:,:,1) = Q(:,:,3);
F(:,:,2) = Q(:,:,2).*v;
F(:,:,3) = Q(:,:,3).*v + P;
F(:,:,4) = (e+P).*v;

Qx_c(J,K,:) = 0.5*(E(JPLUS,K,:) - E(JMINUS,K,:))/dx;
Qy_c(J,K,:) = 0.5*(F(J,KPLUS,:) - F(J,KMINUS,:))/dy;
R = Qx_c + Qy_c;

% $$$   You can add these next 5 lines and run the code w th cfl
% $$$   for 1 revolution to see what a reasonable solutio  would
```

```
 % $$$   I added this in to give you at a sample working c de.
 % $$$   Remove % $$$ from the next lines
 % $$$     epse = 1.0;
 % $$$     Qx_d(J,K,:) = (Q(JPLUS(JPLUS(J)),K,:) - 4*Q(JPL S(J),K
 % $$$         + 6*Q(J,K,:) - 4*Q(JMINUS(J),K,:) + Q(JMINU (JMINU
 % $$$     Qy_d(J,K,:) = (Q(J,KPLUS(KPLUS(K)),:) - 4*Q(J,K LUS(K)
 % $$$         + 6*Q(J,K,:) - 4*Q(J,KMINUS(K),:) + Q(J,KMI US(KMI
 % $$$     R(J,K,:) = R(J,K,:) + epse/12.0*(Qx_d(J,K,:) +  y_d(J,

 resid = norm(R(:,:,1)) /(jmax*kmax);

if resid > 1e5 disp('Residual Exceed 10^5');  break; end

Q = Q - dt*R; % Euler Explicit Step
```

```
  rho_min_error = abs(min(min(Q(:,:,1)))-rho_ex_min);

  if mod(nstep,10) == 0 fprintf(' nstep = %g  Time = %g  Rho Min Er


end

fig2 = figure; %set(gcf,'Position',[950 100 800 800]);
subplot(2,2,1);
surf(X',Y',Q(:,:,1));
xlabel('x');
ylabel('y');
zlabel('rho');
T1 = '2^{nd} Order Central';
T2 = '    Euler Explicit';
```

```
T3 = [ ' \Delta t = ' num2str(dt) '  CFL = ',num2str(cfl) '  M_{\in

suptitle({[ T1 T2]; [T3 ]});

subplot(2,2,2);
contour(X',Y',Q(:,:,1),40);
xlabel('x');
ylabel('y');
zlabel('rho');
grid on;

subplot(2,2,3);
Error(:,:) = Q(:,:,1)-QE(:,:,1);
surf(X',Y',Error);
xlabel('x');
```

```
ylabel('y');
zlabel('rho - rho_{exact}');

subplot(2,2,4);
plot(x,Q(J,floor(kmax/2)+1,1),'ro');
hold on;
plot(x,QE(J,floor(kmax/2)+1,1),'k-');
xlabel('x');
ylabel('rho');
```

Project #11: Assignment

1. Program the Euler Explicit Eq. 22 with the Flux Split, Eq. 24.
2. Use 1^{st},2^{nd} and 3^{rd} order one-sided differences for $\delta_x^b, \delta_x^f, \delta_y^b$, and δ_y^f.
3. Replace the Euler Explicit scheme with Fourth-Order Runge-Kutta and compare the performance and other aspects discussed below.

(a) The RK4 scheme is defined as

$$\begin{aligned}
\widehat{Q}^{n+1/2} &= Q^n + \frac{1}{2}hR(Q)^n \\
\tilde{Q}^{n+1/2} &= Q^n + \frac{1}{2}hR(\widehat{Q})^{n+1/2} \\
\overline{Q}^{n+1} &= Q^n + hR(\tilde{Q})^{n+1/2} \\
Q^{n+1} &= Q^n + \frac{1}{6}h\left[R(Q)^n\right. \\
&\qquad +2\left(R(\widehat{Q})^{n+1/2} + R(\tilde{Q})^{n+1/2}\right) + \\
&\qquad \left. R(\overline{Q})^{n+1}\right] \qquad (25)
\end{aligned}$$

(b) **Note: the proper use of $\widehat{Q}, \tilde{Q}, \overline{Q}$ is very important!**

4. Discuss the spatial accuracy of this method, e.g., what is er_t for the various differences or what is the modified wave number? *Hint* Remember, we are working with a non-linear coupled system. To do the analysis you need to think in terms of the decouple representative equations.

5. Knowing the $\sigma - \lambda$ relation for the Euler explicit and RK4 $O\Delta E$, come up with a stability condition and convergence estimates for different differencing orders. One possible CFL definition for this system is

$$CFL = \frac{h\, max(\lambda(A), \lambda(B))}{\min(dx, dy)}$$

6. Study various CFL numbers (remembering the numerical stability condition!), to see if your analysis is consistent with your results.

7. Pick one:

   (a) Do a grid refinement accuracy study to verify the truncation error for each difference choice, .i.e. $1^{st}, 2^{nd}$ and 3^{rd} Order upwind. Set up a sequence of runs with decreasing grid (increasing nx) and time step sizes. The best way to do this is to intergrate to fixed time, say one revolution. Keep in mind you should keep the CFL number fixed, that is, as you decrease Δx you should correspondingly decrease Δt. Plot the $log(error)$ ($error$ can be defined as the $l2$ norm of the difference between the computed density and the exact density) against $log(\Delta x)$. If the $error \approx O(\Delta x^p)$ then, for example, on a loglog plot a $O(\Delta x^p)$ method should have a

slope of p. Verify this for your results.

(b) Program up at least one more method $O\Delta E$ method and compare it's performance, accuracy and stability with your results above. Look at the other projects for suggestions or just pick one from the textbook or other sources.

i. *Note:* It is not necessary to redo the accuracy and stability analysis for this method, although you may want to make sure it is stable and accurate.)

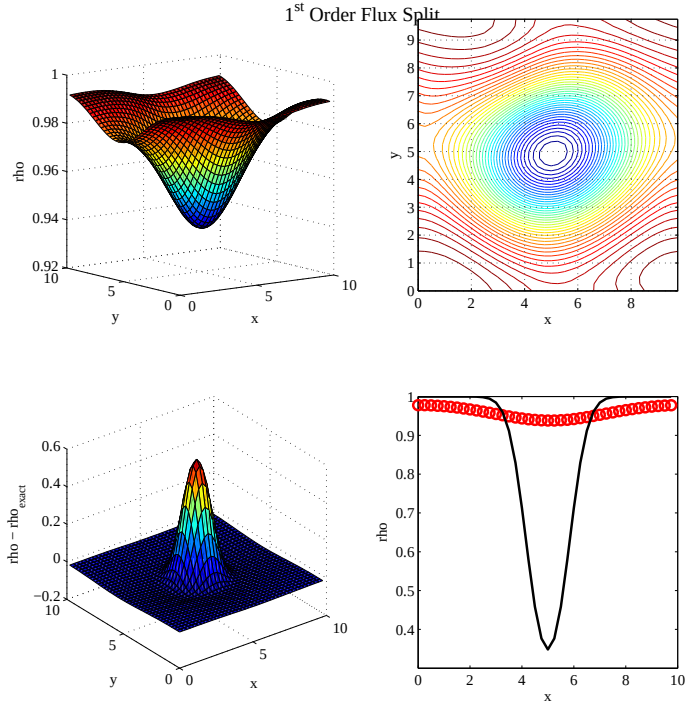
ii. *Suggestion:* Take a look at Lax-Wendroff (see notes). You could even use an unstable scheme, although I prefer a stable one.

(c) Program the Euler Implicit scheme and compare it's efficiency with the Euler explicit results. You can do this with a central difference implicit operator or an LU operator. See the Linear Euler Equation Project #9, for more details.

PROJECT #11: Example

RK4 $\Delta t = 0.1$ CFL = 1.0504 $M_\infty = 0.5$

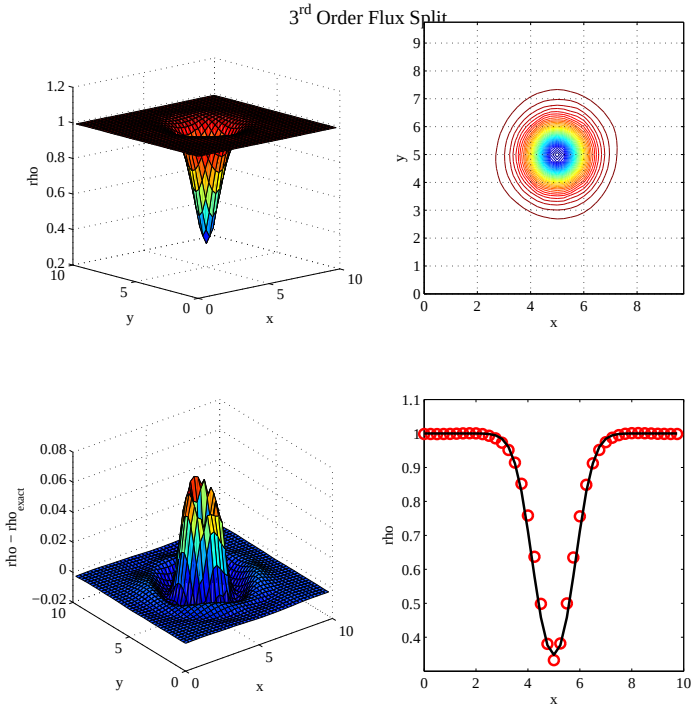
1st Order Flux Split



PROJECT #11: Example

RK4 $\Delta t = 0.1$ CFL = 1.5092 $M_\infty = 0.5$

3rd Order Flux Split



General Instructions

- Follow the instructions given above and address each of the assignments.
- You will need to provide me with a **short** writeup of what you have done, along with some results and figures.
- This can be handwritten, typeset or WORD, pdf file.
- Perform all the computations using MATLAB.
- I want copies of all the source codes by email.
- This project will account for 50% of your grade. I will be judging it on the write-up and a working code (I will run all the codes you send me).